1a

In [39]:
```
N=int(input())
board=[[0]*N for _ in range(N)]
def is_attack(i,j):
    for k in range(N):
        if(board[i][k]==1) or (board[k][j]==1):
            return True

    for k in range(N):
        for l in range(N):
            if(k+l==i+j)or (k-l==i-j):
                if board[k][l]==1:
                    return True
    return False

def q(n):
    if n==0:
        return True

    for i in range(N):
        for j in range(N):
            if(not(is_attack(i,j)) and board[i][j]!=1 ):
                board[i][j]=1
                if q(n-1)==True:
                    return True
                board[i][j]=0
    return False

q(N)
for i in board:
    print(i)
```

```
4
[0, 1, 0, 0]
[0, 0, 0, 1]
[1, 0, 0, 0]
[0, 0, 1, 0]
```

1b

In [ ]:
```
```

1c

In [1]:
```
#bfs
from queue import PriorityQueue
v = 14
graph = [[] for i in range(v)]


def best_first_search(source, target, n):
        visited = [False] * n
        visited[0] = True
        pq = PriorityQueue()
        pq.put((0, source))
        while pq.empty() == False:
                u = pq.get()[1]
                print(u, end=" ")
```

```
                if u == target:
                        break

                for v, c in graph[u]:
                        if visited[v] == False:
                                visited[v] = True
                                pq.put((c, v))
        print()


def addedge(x, y, cost):
        graph[x].append((y, cost))
        graph[y].append((x, cost))

addedge(0, 1, 3)
addedge(0, 2, 6)
addedge(0, 3, 5)
addedge(1, 4, 9)
addedge(1, 5, 8)
addedge(2, 6, 12)
addedge(2, 7, 14)
addedge(3, 8, 7)
addedge(8, 9, 5)
addedge(8, 10, 6)
addedge(9, 11, 1)
addedge(9, 12, 10)
addedge(9, 13, 2)

source = 0
target = 9
best_first_search(source, target, v)
```

```
0 1 3 2 8 9
```

1d

In [2]:

```
from collections import defaultdict

visited=defaultdict(lambda:False)

j1,j2,aim=4,3,2

def w(amt1,amt2):
    if (amt1==2 and amt2==0) or (amt1==0 and amt2==2):
        print(amt1,amt2)
        return True
    if visited[(amt1,amt2)]==False:
        print(amt1,amt2)

        visited[(amt1,amt2)]=True

        return(w(0,amt2) or
               w(amt1,0) or
               w(j1,amt2)or
               w(amt1,j2) or
               w(amt1+min(amt2,j1-amt1),amt2-min(amt2,j1-amt1))or
               w(amt1-min(amt1,j2-amt2),amt2+min(amt1,j2-amt2))
        )
    else:
        return False

w(0,0)
```

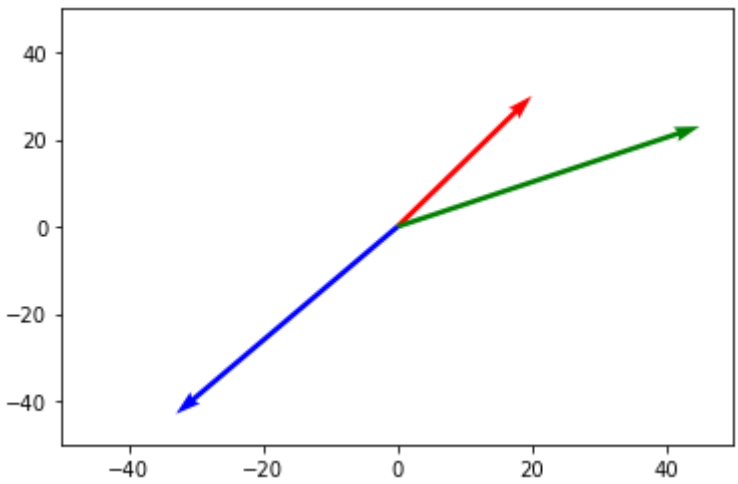
```
0 0
4 0
4 3
0 3
3 0
3 3
4 2
0 2
```

Out[2]: True

2 2d AND 3D

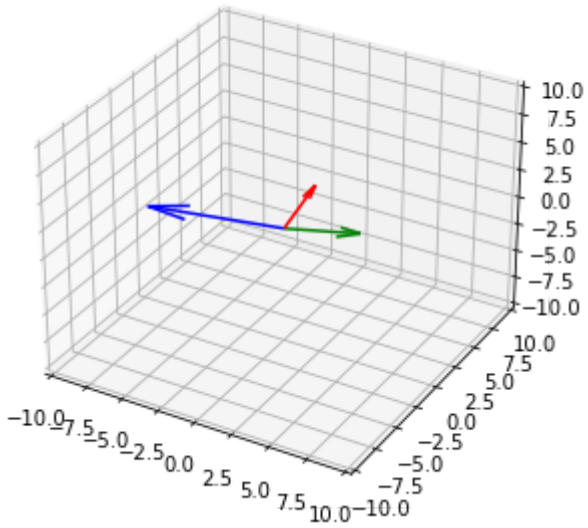
In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

def plt2d(v,color):
    origin=[0,0]
    plt.quiver(*origin,*v,color=color,angles='xy',scale_units='xy',scale=1)

def plt3d(ax,v,color):
    ax.quiver(*[0,0,0],*v,color=color)
```

In [4]:
```
#2d
x=[20,30]
y=[45,23]
z=[-33,-43]

plt.xlim(-50,50)
plt.ylim(-50,50)

plt2d(x,'r')
plt2d(y,'g')
plt2d(z,'b')
```



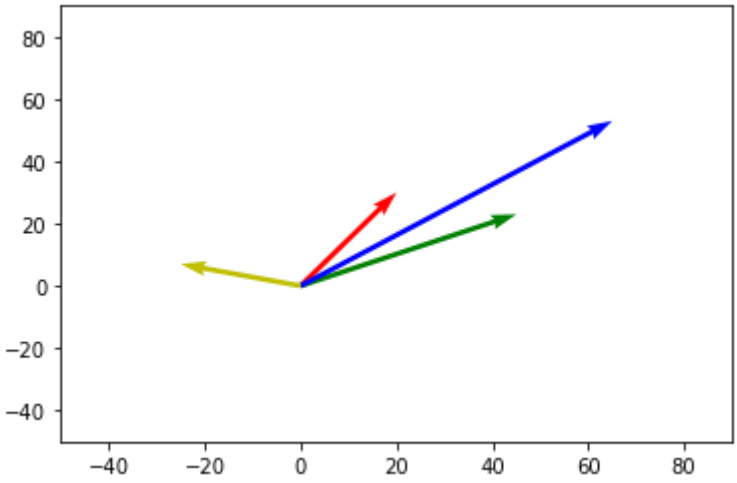
In [5]:
```
#3d

v1=[1, 2, 3]
v2=[-7, -4, 2]
v3=[3, 4, -2]

fig=plt.figure(figsize=(5,5))
ax=plt.axes(projection='3d')
```

```
plt.xlim([-10,10])
plt.ylim([-10,10])
ax.set_zlim([-10,10])

plt3d(ax,v1,'r')
plt3d(ax,v2,'b')
plt3d(ax,v3,'g')
```



In [6]:

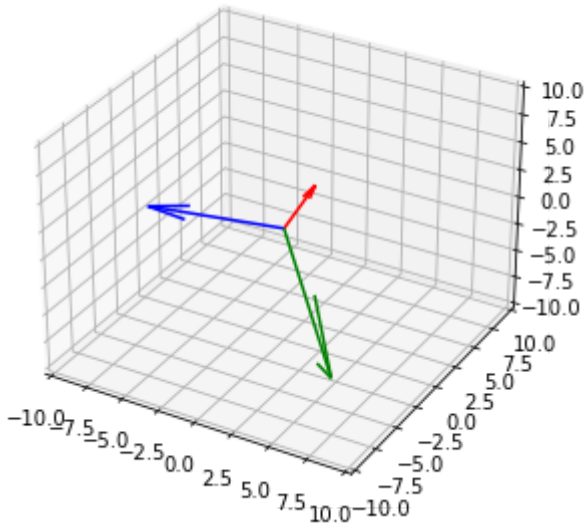
```
#vector addition
x1=np.array([20,30])
y1=np.array([45,23])

plt.xlim([-50,90])
plt.ylim([-50,90])

plt2d(x1,'r')
plt2d(y1,'g')
plt2d(x1+y1,'b')

#subtraction
plt2d(x1-y1,'y')
```



In [7]:

```
#cross product
j1=[1, 2, 3]
j2=[-7, -4, 2]

fig=plt.figure(figsize=(5,5))
```

```
ax=plt.axes(projection='3d')


plt.xlim([-10,10])
plt.ylim([-10,10])
ax.set_zlim([-10,10])

plt3d(ax,j1,'r')
plt3d(ax,j2,'b')
plt3d(ax,np.cross(j1,j2),'g')
```



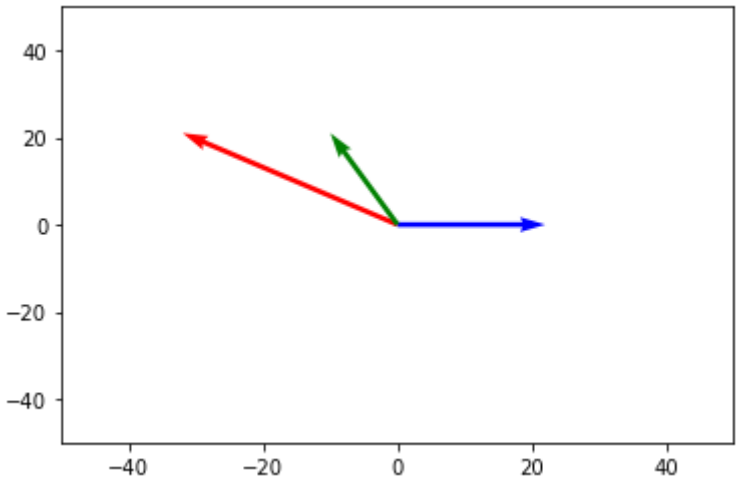
3

In [37]:

```
a=np.array([-32,21])
b=np.array([22,0])
maga=np.linalg.norm(a)
magb=np.linalg.norm(b)
dot=np.dot(a,b)
s=np.add(a,b)

if (maga*magb>=dot):
    print('sch')
else:
    print('no sch')
if (np.add(maga,magb)>=np.linalg.norm(s)):
    print('tri')
else:
    print('no tri')

plt.xlim([-50,50])
plt.ylim([-50,50])

plt.quiver(*[0,0],*a,angles='xy',scale=1,scale_units='xy',color='r')
plt.quiver(*[0,0],*b,angles='xy',scale=1,scale_units='xy',color='b')
plt.quiver(*[0,0],*(a+b),angles='xy',scale=1,scale_units='xy',color='g')
```

```
sch
tri
```

Out[37]: <matplotlib.quiver.Quiver at 0x1ce37c0dd60>

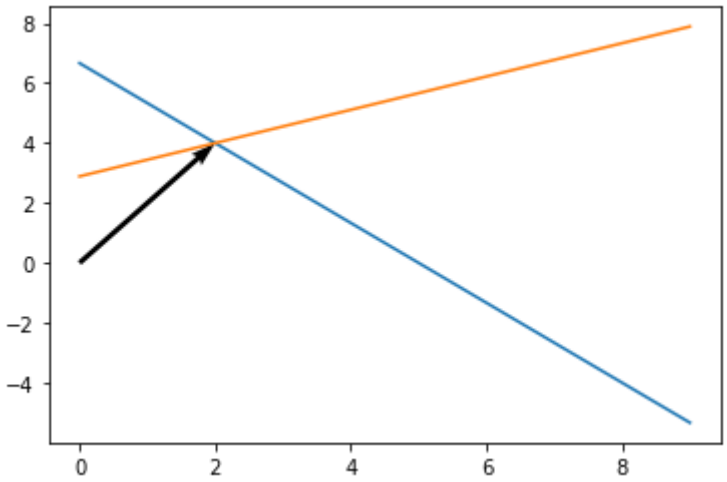


4

In [38]:
```
import matplotlib.pyplot as plt
import numpy as np
a=np.array([[4,3],[-5,9]])
b=np.array([20,26])
print('inverse=',np.linalg.inv(a))
e=np.linalg.solve(a,b)
print('the sol',e)
xp=np.linspace(0,9,1000)
y1=(20-4*xp)/3
y2=(26+5*xp)/9
plt.plot(xp,y1)
plt.plot(xp,y2)
plt.quiver(*[0,0],*[2,4],scale=1,scale_units="xy",angles='xy',units='xy')
```

```
inverse= [[ 0.17647059 -0.05882353]
 [ 0.09803922  0.07843137]]
the sol [2. 4.]
```

Out[38]: <matplotlib.quiver.Quiver at 0x1ce38c38910>



In [ ]:

5

In [9]:
```
import pandas as pd
import numpy as np
```

```
import seaborn as sns
s=sns.load_dataset('iris')
s.to_csv(r"C:\Users\kisha\Desktop\k.csv")

data=pd.read_csv(r"C:\Users\kisha\Desktop\k.csv")
data.head()
```

Out[9]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
0	0	5.1	3.5	1.4	0.2	setosa
1	1	4.9	3.0	1.4	0.2	setosa
2	2	4.7	3.2	1.3	0.2	setosa
3	3	4.6	3.1	1.5	0.2	setosa
4	4	5.0	3.6	1.4	0.2	setosa

In [10]:

```
data.count()
```

Out[10]:

```
Unnamed: 0      150
sepal_length    150
sepal_width     150
petal_length    150
petal_width     150
species         150
dtype: int64
```

In [11]:

```
data.value_counts('sepal_length')
```

Out[11]:

```
sepal_length
5.0    10
6.3     9
5.1     9
5.7     8
6.7     8
6.4     7
5.8     7
5.5     7
5.4     6
6.1     6
5.6     6
6.0     6
4.9     6
4.8     5
6.5     5
6.2     4
5.2     4
4.6     4
7.7     4
6.9     4
7.2     3
5.9     3
4.4     3
6.8     3
4.7     2
6.6     2
7.4     1
7.6     1
7.3     1
4.3     1
7.1     1
7.0     1
5.3     1
```

```
4.5     1
7.9     1
dtype: int64
```

In [12]:
```
data.sort_values(by='sepal_length',inplace=False,axis=0,ascending=False)
```

Out[12]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
131	131	7.9	3.8	6.4	2.0	virginica
135	135	7.7	3.0	6.1	2.3	virginica
122	122	7.7	2.8	6.7	2.0	virginica
117	117	7.7	3.8	6.7	2.2	virginica
118	118	7.7	2.6	6.9	2.3	virginica
...	...	...	...	...	...	...
41	41	4.5	2.3	1.3	0.3	setosa
42	42	4.4	3.2	1.3	0.2	setosa
38	38	4.4	3.0	1.3	0.2	setosa
8	8	4.4	2.9	1.4	0.2	setosa
13	13	4.3	3.0	1.1	0.1	setosa

150 rows × 6 columns

In [13]:
```
data.iloc[1,2]=float("NaN")
data.head()
data.dropna(inplace=False)
```

Out[13]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
0	0	5.1	3.5	1.4	0.2	setosa
2	2	4.7	3.2	1.3	0.2	setosa
3	3	4.6	3.1	1.5	0.2	setosa
4	4	5.0	3.6	1.4	0.2	setosa
5	5	5.4	3.9	1.7	0.4	setosa
...	...	...	...	...	...	...
145	145	6.7	3.0	5.2	2.3	virginica
146	146	6.3	2.5	5.0	1.9	virginica
147	147	6.5	3.0	5.2	2.0	virginica
148	148	6.2	3.4	5.4	2.3	virginica
149	149	5.9	3.0	5.1	1.8	virginica

149 rows × 6 columns

In [14]:
```
data.iloc[4,2]
```

Out[14]: 3.6

In [15]:
```
data.describe()
```

Out[15]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	149.000000	150.000000	150.000000
mean	74.500000	5.843333	3.057718	3.758000	1.199333
std	43.445368	0.828066	0.437311	1.765298	0.762238
min	0.000000	4.300000	2.000000	1.000000	0.100000
25%	37.250000	5.100000	2.800000	1.600000	0.300000
50%	74.500000	5.800000	3.000000	4.350000	1.300000
75%	111.750000	6.400000	3.300000	5.100000	1.800000
max	149.000000	7.900000	4.400000	6.900000	2.500000

In [16]:
```
type(data)
```

Out[16]: pandas.core.frame.DataFrame

In [17]:
```
data.loc[(data.sepal_length)>3]
```

Out[17]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
0	0	5.1	3.5	1.4	0.2	setosa
1	1	4.9	NaN	1.4	0.2	setosa
2	2	4.7	3.2	1.3	0.2	setosa
3	3	4.6	3.1	1.5	0.2	setosa
4	4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...	...
145	145	6.7	3.0	5.2	2.3	virginica
146	146	6.3	2.5	5.0	1.9	virginica
147	147	6.5	3.0	5.2	2.0	virginica
148	148	6.2	3.4	5.4	2.3	virginica
149	149	5.9	3.0	5.1	1.8	virginica

150 rows × 6 columns

In [18]:
```
data.drop(["sepal_length"],axis=1)
```

Out[18]:

	Unnamed: 0	sepal_width	petal_length	petal_width	species
0	0	3.5	1.4	0.2	setosa
1	1	NaN	1.4	0.2	setosa

	Unnamed: 0	sepal_width	petal_length	petal_width	species
2	2	3.2	1.3	0.2	setosa
3	3	3.1	1.5	0.2	setosa
4	4	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	145	3.0	5.2	2.3	virginica
146	146	2.5	5.0	1.9	virginica
147	147	3.0	5.2	2.0	virginica
148	148	3.4	5.4	2.3	virginica
149	149	3.0	5.1	1.8	virginica

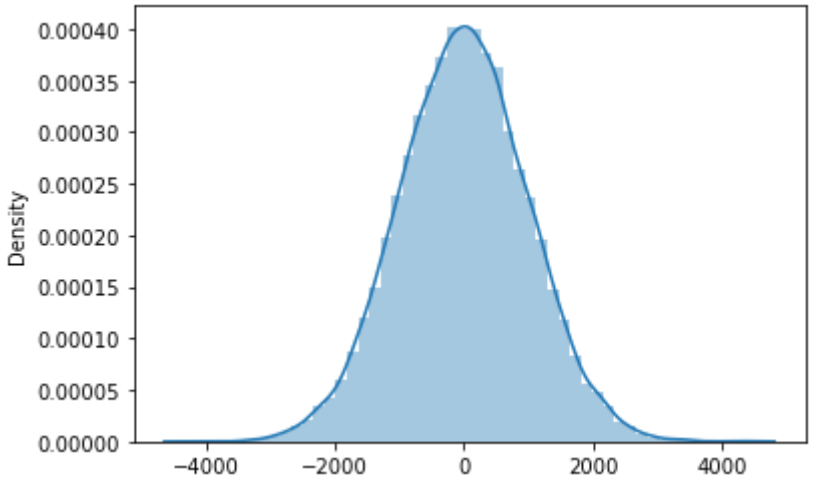
150 rows × 5 columns

6

In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

sns.distplot(np.random.normal(loc=0,scale=1000,size=20000),kde=True)
```

```
C:\Users\kisha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

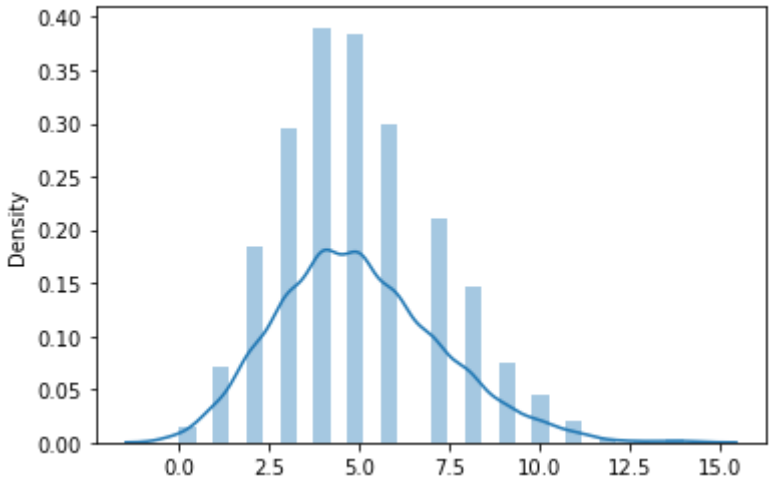
Out[19]: <AxesSubplot:ylabel='Density'>



In [20]:
```
sns.distplot(np.random.poisson(lam=5,size=2000),kde=True)
```

```
C:\Users\kisha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

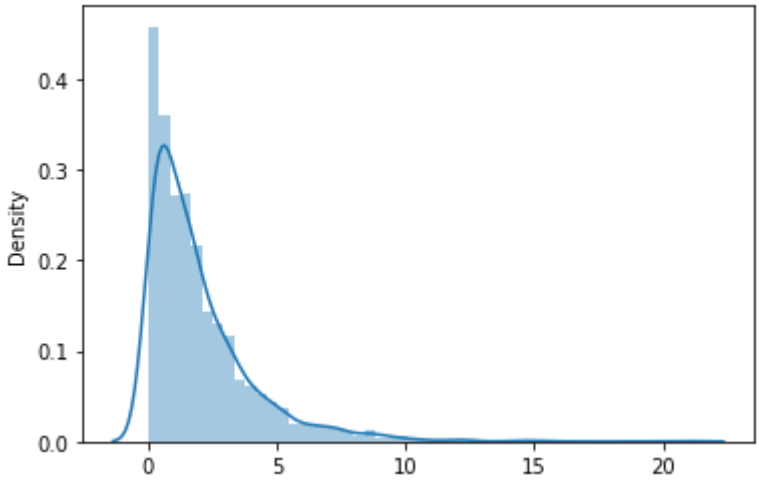
Out[20]: <AxesSubplot:ylabel='Density'>



In [21]:
```
sns.distplot(np.random.chisquare(df=2,size=2000),kde=True,hist=True)
```

```
C:\Users\kisha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

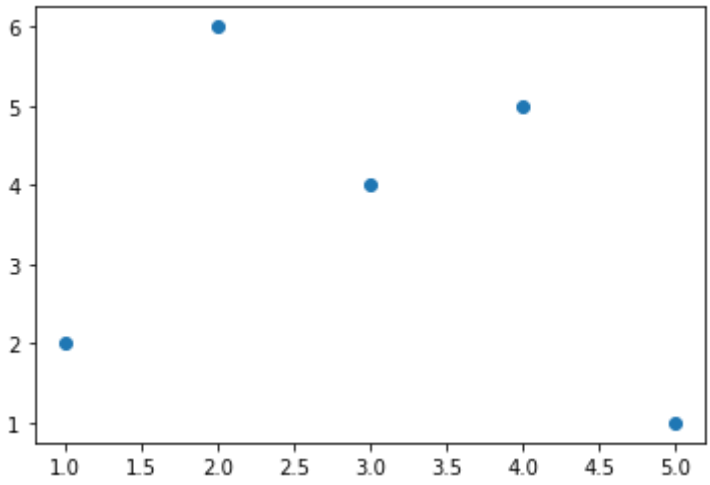
Out[21]: <AxesSubplot:ylabel='Density'>



7

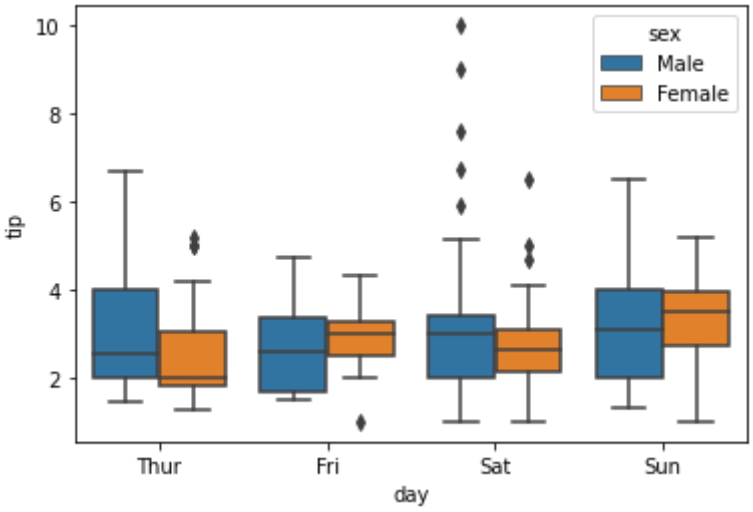
In [22]:
```
#scatter
x=[1,3,4,5,2]
y=[2,4,5,1,6]
plt.scatter(x,y)
```

Out[22]: <matplotlib.collections.PathCollection at 0x1ce367af580>



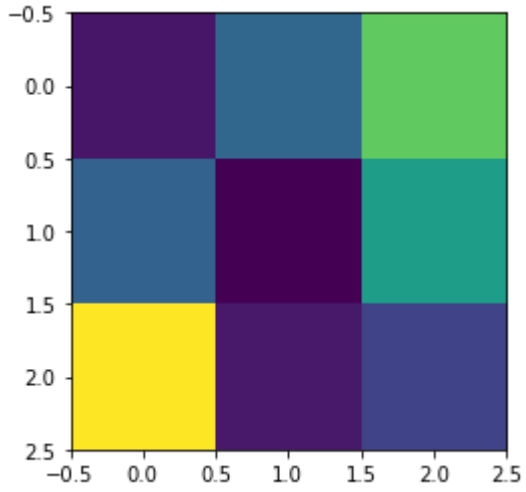
In [23]:
```
#box
dataa=sns.load_dataset('tips')
sns.boxplot(x='day',y='tip',data=dataa,hue='sex')
```

Out[23]: <AxesSubplot:xlabel='day', ylabel='tip'>



In [24]:
```
j=np.random.rand(3,3)
plt.imshow(j)
```

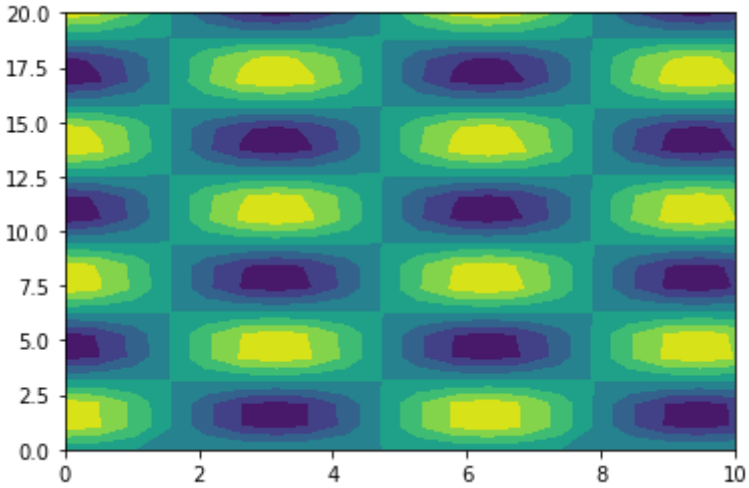
Out[24]: <matplotlib.image.AxesImage at 0x1ce368ffee0>



In [25]:
```
f=lambda x,y:np.cos(x)*np.sin(y)
```

```
x=np.linspace(0,10,20)
y=np.linspace(0,20,20)
X,Y=np.meshgrid(x,y)
Z=f(X,Y)
plt.contourf(X,Y,Z)
```

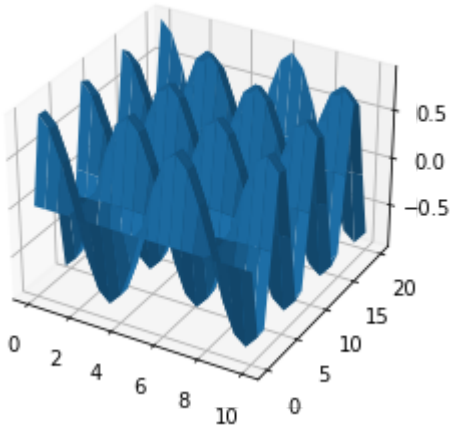
Out[25]: <matplotlib.contour.QuadContourSet at 0x1ce36966b50>



In [26]:

```
import mpl_toolkits.mplot3d
f=lambda x,y:np.cos(x)*np.sin(y)
x=np.linspace(0,10,20)
y=np.linspace(0,20,20)
X,Y=np.meshgrid(x,y)
Z=f(X,Y)
fig=plt.figure()
ax=fig.gca(projection="3d")
ax.plot_surface(X,Y,Z)
```

Out[26]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x1ce36a0c340>



In [27]:

```
import numpy as np

n=np.random.normal(size=1000)
print(np.var(n),
np.std(n),
np.average(n))

n=np.random.chisquare(df=10,size=100)
print(np.var(n),
np.std(n),
```

```
np.average(n))

n=np.random.exponential(size=1000)
print(np.var(n),
np.std(n),
np.average(n))

n=np.random.poisson(lam=20,size=10)
print(np.var(n),
np.std(n),
np.average(n))
```

```
0.9927796116810009 0.99638326545612 -0.009190257438595594
21.943246487130768 4.684361908214476 10.828357100899753
0.9987624724915642 0.9993810446929461 0.9890745510112481
38.61 6.2136945531623935 19.3
```

8

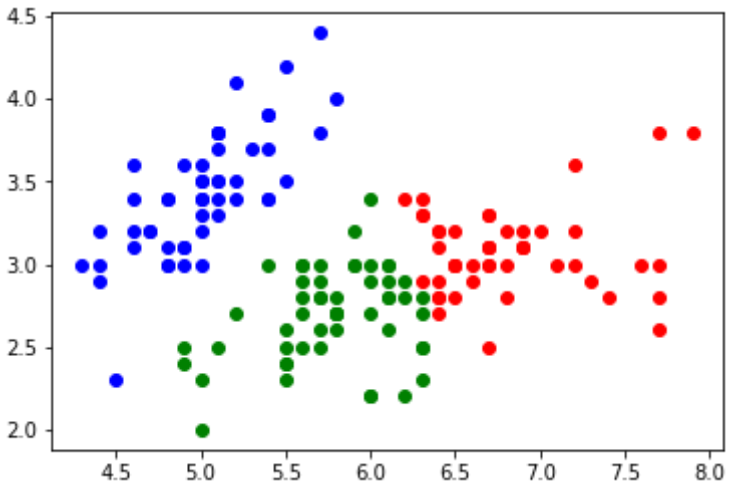
In [28]:
```
#8a
from sklearn.cluster import KMeans

j=sns.load_dataset('iris')
k=j.drop(['petal_length',"petal_width","species"],axis=1)
```

In [29]:
```
km=KMeans(n_clusters=3)
y=km.fit_predict(k)
```

In [30]:
```
k['c']=y
```

In [31]:
```
k1=k[k['c']==0]
k2=k[k['c']==1]
k3=k[k['c']==2]

plt.scatter(k1.sepal_length,k1.sepal_width,color='r')
plt.scatter(k2.sepal_length,k2.sepal_width,color='b')
plt.scatter(k3.sepal_length,k3.sepal_width,color='g')
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x1ce37ab00a0>
```



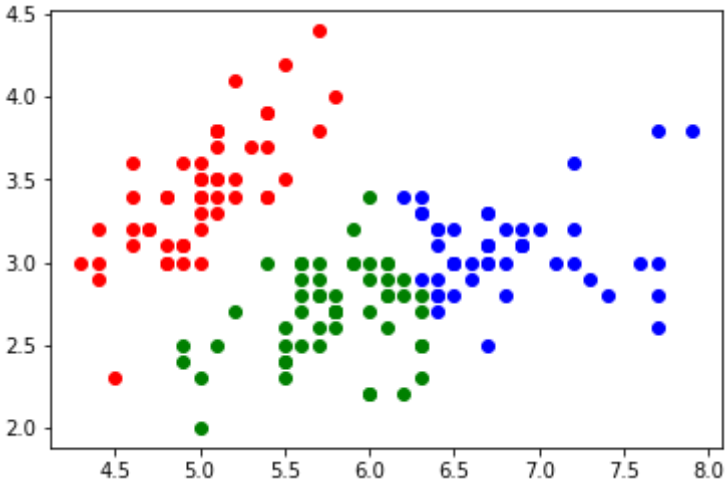
In [32]:
```
from sklearn.mixture import GaussianMixture as g
```

```
km1=g(n_components=3)
y1=km1.fit_predict(k)
```

In [33]:

```
k['c1']=y1
k1=k[k['c1']==0]
k2=k[k['c1']==1]
k3=k[k['c1']==2]

plt.scatter(k1.sepal_length,k1.sepal_width,color='r')
plt.scatter(k2.sepal_length,k2.sepal_width,color='b')
plt.scatter(k3.sepal_length,k3.sepal_width,color='g')
```

Out[33]: <matplotlib.collections.PathCollection at 0x1ce337d7c40>



In [ ]: